ETRI
전자통신동향분석 제23권 제1호 2008년 2월

글로벌 모바일 단말 소프트웨어 플랫폼 동향

Global Mobile Software Platform Trends

임베디드 S/W 기술 동향 특집

윤민홍 (M.H. Yun) 리눅스모바일단말SW연구팀 연구원
김선자 (S.J. Kim) 리눅스모바일단말SW연구팀 연구원

목 차

모바일 플랫폼은 단말기에 탑재되어 단말기의 하드웨어 기능을 상위 계층에서 사용할 수 있도록 하여 주고, 상위 응용 계층에는 프로그래밍 환경 및 실행 환경을 제공하는 역할을 한다. CDMA 진영에서는 운영체제와 구분되어 사용되기도 하나 단말기 하드웨어의 성능이 향상되고 다양한 기능이 요구되어 운영체제로부터 애플리케이션 프레임워크까지의 모든 소프트웨어를 일컫는 말로 사용되고 있다. WIPI, BREW, J2ME, Symbian 등 기존의 모바일 플랫폼 영역에 2007년에는 애플의 새로운 스마트폰인 iPhone과 구글의 새로운 플랫폼인 Android가 등장하여 모바일 플랫폼 경쟁은 더욱 심화되고 있다. 본 논문에서는 모바일 플랫폼 중 비중이 급격히 확대되고 있는 스마트폰용 모바일 플랫폼의 동향을 분석하고, 국내외의 모바일 플랫폼 동향을 예측해 본다.

Ⅰ. 모바일 소프트웨어 플랫폼

과거 스마트폰은 PDA 폰이라는 이름의 PDA 기능과 폰의 기능이 물리적으로 합쳐진 단말에서 시작하였다. 반면 최근의 스마트폰은 폰 중심의 기능이 PDA 기능을 흡수한 양상을 보이는데, 이를 통해 두 기능의 유기적 동작이 가능해졌다[1],[2].

모바일 플랫폼은 스마트폰뿐 아니라 기본 기능만 제공하는 폰(vanilla phone)부터 시작하여 카메라 폰 또는 MP3 폰과 같은 기능폰(feature phone)에도 탑재되는 소프트웨어로, 폰의 하드웨어적 성능을 상위 응용소프트웨어에서 사용할 수 있도록 하는 역할을 한다. 휴대 단말에 탑재된 운영체제와 플랫폼을 구분하기도 하나 범용 운영체제가 사용되면서 운영체제부터 애플리케이션 프레임워크까지의 모든 소프트웨어를 일컫는 말로 사용되고 있다. 한 예로, Qualcomm의 CDMA 기술을 사용하며 MSM 칩을 주로 사용하는 한국의 경우 동사의 REX를 운영체제로 사용하고, 이 위에 무선인터넷 서비스를 위한 WIPI 등을 플랫폼으로 탑재하여 사용했다. 모바일 플랫폼은 모바일 단말에 탑재되는 소프트웨어로 상위 응용프로그램이 다양한 서비스를 제공할 수 있도록 하는 데 도움을 주는 소프트웨어를 말한다.

Apple은 휴대 단말기인 iPhone을 2007년 1월 9일 발표했다. Google은 2008년 출시 계획을 가진 구글폰(gPhone)의 플랫폼인 Android의 SDK를 2007년 11월 12일 공개했다. Apple의 iPhone은 차별화된 UI를 통해 매니아층을 형성시켜 iPod에 이어 Apple의 휴대 단말의 성공을 이끌어 냈다. Google의 Android는 인터넷 서비스 업체인 Google이 만들어낸 개방형 플랫폼으로 제품 출시 전부터 SDK를 공개하여 많은 개발자를 확보해 나가고 있다. 기존의 플랫폼들은 범용 운영체제가 지원되기 전의 휴대폰부터 진화를 해나가 불필요한 기능을 가지고 있는 등의 단점이 있었으나, 이 두 플랫폼은 스마트폰에서 시작하여 이런 단점을 가지고 있지 않다. WIPI, BREW, J2ME, Symbian, Windows Mobile 등 기존의 모바일 플랫폼 영역에 두 플랫폼이 새로 등장하여 모바일 플랫폼 경쟁은 더욱 심화되었다.

본 논문에서는 모바일 플랫폼 중 비중이 급격히 확대되고 있는 스마트폰용 모바일 플랫폼의 동향을 분석하고, 국내외의 모바일 플랫폼 동향을 예측해 본다.

● 용 어 해 설 ●

모바일 플랫폼: 모바일 단말의 기반이 되는 소프트웨어. 모바일 단말은 하드웨어 계층, 펌웨어 및 운영체제 계층으로 차례로 구성된다. 다양한 응용 프로그램을 제공하기 위해 운영체제 위에 응용프로그램을 위한 계층을 두기도 한다. 모바일 플랫폼은 다양한 응용프로그램을 제공하기 위해 모바일 단말에 탑재된 소프트웨어 계층들을 총칭한다.

WIPI(Wireless Internet Platform for Interoperability): 한국무선인터넷 표준화 포럼(KWISF: Korea Wireless Internet Standardization Forum)의 모바일 플랫폼 특별 분과에서 표준화한 모바일 플랫폼 표준 규격. 무선인터넷을 통해 다운로드한 응용프로그램을 이동통신 단말기에 탑재시켜 실행시키기 위한 환경을 제공하는 데 필요한 표준규격이다.

Ⅱ. 스마트폰용 모바일 플랫폼 고려사항

1. 프로세서 구조

스마트폰용 모바일 단말의 프로세서 구조는 응용 프로세서 부분과 베이스밴드 프로세서 부분이 독립되어 있는 듀얼 프로세서 구조가 일반적이었다. 이러한 구조에서는 운영체제 및 응용 소프트웨어는 응용 프로세서에서 실행되며, 베이스밴드용 프로토콜 처리 등에 대해서는 응용 프로세서가 관여하지 않는 방식이 사용된다. 이런 듀얼 프로세서 구조에서는 베이스밴드 프로세서를 주변장치로 인식하여 통신 처리를 진행한다[3].

듀얼 프로세서 구조를 갖게 되는 스마트폰용 프로세서의 예로는 인텔의 PXA25x, PXA27x 등과 TI의 OMAP15xx, OMAP16xx 등, 그리고 모토롤라의 i.MX1, 삼성전자의 ARM9 기반 SoC 등이 있

다. (그림 1)에서 보는 바와 같이 이들 프로세서는 UART, USB, 혹은 특별한 인터페이스를 사용해 베이스밴드 프로세서와 연결되고 이를 제어하는 방식으로 통신 기능을 구현한다. 듀얼 프로세서 구조의 장점은 복잡한 통신 프로토콜에 대해 신경쓸 필요 없이 응용프로세서 부분에서 응용 소프트웨어 실행만 하면 되고, 통신 부분은 비교적 간단하게 해결할 수 있다는 것이다. 그러나, 통신 처리시 발생할 수 있는 다양한 시나리오를 완벽하게 처리할 수 없는 경우가 발생해 베이스밴드 부분의 통신 프로토콜 처리 소프트웨어를 수정해야 하는 경우가 발생한다는 단점이 있다.

(그림 1)의 듀얼 프로세서 기반의 스마트폰 구조에서는 베이스밴드 프로세서부의 control firmware는 통신 프로토콜 처리 소프트웨어에 맞는 통신 제어 소프트웨어 모듈이 탑재되며, 이는 음성 통신 및 데이터 통신을 처리하게 된다.

이에 반해 싱글 프로세서 구조에서는 하나의 프로세서 내에 두 개 이상의 코어를 갖추고 있어 운영체제는 두 개의 코어를 동시에 제어할 수 있는 기능을 가져야 한다[3]. 대표적인 싱글 프로세서 멀티코어 구조의 SoC로는 TI의 OMAP 프로세서와 Qualcomm의 7500 시리즈가 있다. 이러한 구조의

(그림 1) 듀얼 프로세서 구조 스마트 폰

프로세서에서 가장 중요한 소프트웨어 계층은 서로 다른 코어 사이의 데이터 교환을 가능하게 해주는 인터페이스 계층이다. TI의 OMAP 프로세서의 경우에는 응용 프로세서 코어로 ARM9을 사용하고 TI의 DSP 코어인 C55x 코어를 사용하여 멀티미디어 처리나 통신 프로토콜 처리를 한다. TI에서는 응용 프로세서 코어인 ARM9 코어와 DSP 사이의 인터페이스를 위해 (그림 2)에서 보는 바와 같이 DSP/BIOS Bridge라고 하는 소프트웨어 계층을 주요 운영체제에 대해 제공한다[4].

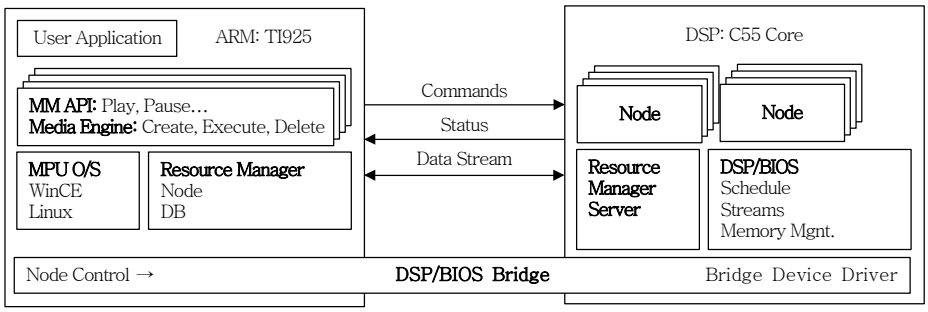
2. 시스템 지원 기술

가. 전력 관리 기술

모바일 단말기에서 전력 소모를 줄이기 위한 방안으로 소프트웨어 수준에서의 전력 관리 기술을 고려해야 한다. 동적 CPU 클럭 속도 변경(dynamic CPU frequency scaling)이나 동적인 운용 전압 변경(dynamic voltage scaling) 등과 같은 기술의 적용을 위해 많은 노력이 필요하다[5].

배터리 충전 소프트웨어에서 중요한 요소는 배터리 충전 알고리듬이다. 지속적으로 현재 배터리 전압 상태를 파악하여 안정적인 전압 상태가 될 때까지 계속 충전을 해야 한다. 충전시 비정상적인 상황이나 예상치 못한 상황이 생길 것을 충분히 대비한 알고리듬을 구현해야 한다. 운영체제 수준에서의 전력 관리는 전력 관리를 위한 기본 플랫폼과 전력 관리 전략 구현으로 나눌 수 있다.

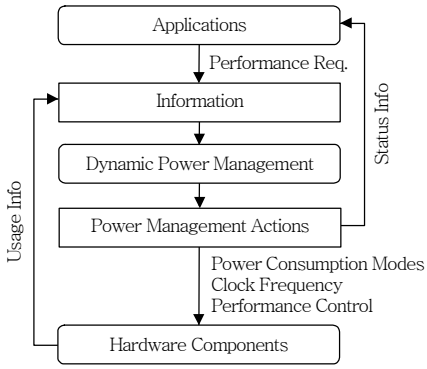
전력관리를 위한 기본 플랫폼은 전력 관리의 전



(그림 2) TI 프로세서를 위한 DSP/BIOS Bridge

략적 부분의 구현에 필요한 인터페이스 및 장치 제어부를 말한다. (그림 3)에서 보는 바와 같이 기본 플랫폼에는 전력 관리 가능한 장치의 장치 드라이버, CPU의 전력 소모 모드 전환을 위한 인터페이스, 전력 관리 기능을 응용 소프트웨어에서 사용할 수 있도록 하는 인터페이스, 그리고 전력 관리 전략을 구현하기 위한 기본 인터페이스 등이 포함된다. 전력 소모를 관리할 수 있는 요소로는 그림에 나타나 있듯이 각 장치의 전력 소모 모드, CPU의 구동 클럭 속도, 그리고 메모리 버스 속도 등 성능에 영향을 줄 수 있는 요소들이다.

전력 관리 전략 구현은 현재 간단한 수준을 벗어나지 못하고 있다. 단순히 일정 시간 동안 각 장치들이 사용되지 않는 경우 각 장치의 전력 소모 모드를 저전력 모드로 전환시키고, 특히 CPU가 수행할 작업이 일정 시간 없을 경우 바로 idle 모드 또는 sleep 모드로 CPU를 전환시키는 정도이다. 인텔에서 권장하는 전력 소모 관리 전략 중 현재 시스템에서 수행하는 작업을 계산량이 많은 작업에 해당하는지 또는 메모리 접근이 주가 되는 작업에 해당하는지 분류하여 상황에 맞게 시스템 설정을 변경하는 것이 있다. 계산량이 많은 작업으로 판단되는 경우에는 메모리 버스 속도는 낮추되 CPU 속도를 높여 작업을 빨리 끝내는 방향으로 유도하고, 메모리 접근이 많고 전송 데이터의 양이 많은 작업의 경우에는 CPU 속도는 최대한 낮추되 메모리 접근 속도를 최대한 높이는 방향으로 시스템 설정을 변경하여 전력 소모를 줄인다.



(그림 3) 전력 관리 기본 개념

나. Fast Boot 기술

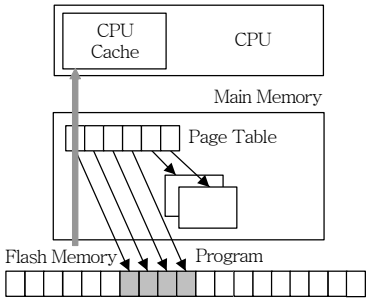
단말의 부팅 시간에 영향을 주는 요소는 실행 메모리의 속도, 파일 시스템의 구성, 부팅에 필요한 시스템 응용들의 실행 속도, 부팅에 필요한 시스템 응용들의 실행 순서 등이다. 이 기술은 커널 XIP 기술 등을 개선하고, 많은 실험에 의해 개선해야 한다.

다. XIP 기술

XIP는 플래시 메모리에서 응용 프로그램을 직접 실행하는 방법으로 이 방법을 사용하면 응용 프로그램 바이너리를 DRAM으로 복사하는 오버헤드를 없애고 전력 소모를 줄일 수 있다는 장점을 가질 수 있다. 그러나 XIP를 사용하기 위해선 파일 시스템 내에 응용 프로그램의 바이너리가 순차적으로 저장되어 있어야 하기 때문에, 일반적으로 사용되는 플래시 메모리용 파일 시스템 상에서는 XIP를 구현할 수 없다[6].

XIP는 커널 XIP와 사용자 응용 프로그램의 XIP로 나눌 수 있는데 사용자 응용 프로그램의 XIP 기능에 대해서만 어느 정도 안정화된 솔루션이 존재하는 수준이다. 사용자 응용 프로그램의 XIP를 구현한 예는 (그림 4)와 같다.

(그림 4)에서와 같이 XIP는 page fault 발생시 메인 메모리에 페이지를 복사하는 1차 적재 과정을 없애고 바로 CPU가 실행할 수 있도록 페이지의 주소를 플래시 주소로 매핑시킨다. 이런 과정에 의해 메인 메모리를 거치지 않고 바로 CPU의 캐시로 페이지 내용이 적재된다. 이를 구현하기 위해서는 page



(그림 4) XIP 프로그램 실행 예

fault 발생시 이를 처리하는 알고리듬이 파일 시스템 코드와 커널 내의 메모리 관리 코드 부분에 구현되어야 한다.

라. 멀티미디어 처리 기술

휴대 단말에서 멀티미디어 기능은 중요한 기능 중 하나다. MP3는 물론 동영상뿐 아니라 DMB도 지원해야 하며 음성 및 동영상 캡처도 지원해야 한다. 동영상 캡처를 위해서 카메라 디바이스 드라이버, 동영상 입력 데이터의 전송을 위한 DMA, 캡처와 동시에 디스플레이를 하기 위한 디스플레이 부분 등을 고려해야 한다.

동영상 데이터를 재생하기 위해서는 인코딩시 수행했던 과정을 역으로 수행해야 한다. 각 단계의 최적화도 고려해야 하며, 디스플레이를 위한 버퍼의 관리도 고려해야 한다.

디코더 및 플레이어가 관리하는 디코더 버퍼와 플레이어 버퍼가 있으며, LCD 출력을 위해 사용하는 마지막 단계의 프레임 버퍼가 존재한다. 디코더 버퍼, 플레이어 버퍼, 그리고 프레임 버퍼를 모두 따로 유지하는 방법이 있으며, 이들을 공유하는 방법이 있다. 각 버퍼를 별도로 유지하는 방법은 구현이 용이하나 두 번의 버퍼간 복사 작업이 필요하다는 단점이 있다.

버퍼를 공유하는 방법에선 디코딩한 결과로 나온 영상 데이터를 플레이어가 바로 사용한다. 이로써 영상 데이터의 복사 작업을 줄일 수 있다.

마. OOM 처리 기술

OOM이란 잔여 메모리 용량이 지극히 부족한 상황을 말하는 것으로 휴대 단말기에서는 이러한 상황을 특별하게 처리해야 한다. 잔여 메모리 용량이 부족하여 새로운 프로세스 진행이 불가능할 경우 이미 실행중인 프로세스 중 중요도가 낮은 프로세스를 골라 실행을 중지시켜 잔여 메모리를 확보한 후 새로운 작업을 개시하는 방식으로 OOM 상황을 처리하는 것이 일반적이다. Symbian에서는 OOM 처리 부분을 따로 정의하고 있어 사용자가 OOM 상황을 어떻게 처리할지 명시할 수 있다. 이차 저장장치인 플래시 메모리의 접근 속도가 현저히 느려 일반적인 방식인 SWAP 영역을 이차 저장 장치에 구현하는 것이 사실한 불가능하기 때문에 OOM 처리 기술이 필요하다.

바. GUI 플랫폼 및 미들웨어 기술

GUI의 요소 중 하나인 윈도 매니저, 혹은 응용 프로그램 런처(launcher)는 일반 데스크톱 시스템에서의 GUI와 달리 시스템의 각종 정보를 받아 들이고 이에 대한 처리를 담당해야 한다. 예를 들어 통신 상황이나 배터리 상황 정보를 받아 화면에 표시해야 한다. 따라서 휴대 단말용 GUI 시스템에선 이러한 시스템의 정보를 수집하는 메커니즘에 대한 정의가 필요하다.

오랫동안 널리 사용되어온 Symbian은 다양하면서도 완성도가 높은 미들웨어를 제공하여 휴대 단말상에서의 서비스가 빠르게 구현될 수 있다는 장점을 가지고 있다. 이처럼 휴대 단말용 플랫폼은 다양한 이동통신 서비스를 지원하기 위해 풍부한 미들웨어를 갖추어야 한다. 또 각 미들웨어는 새로운 표준이나 기능의 추가를 수월하게 하는 유연성을 갖추어야 한다.

사. 기타 고려사항

NAND 플래시 메모리를 위한 파일 시스템 기술이 필요하다. 일부에서는 NOR 플래시 메모리 환경을 가정하고 개발된 파일 시스템을 NAND 플래시 메모리에 그대로 사용하기도 하나 성능상의 낭비가 불가피하다. 또한, 요구 페이징(demand paging) 부분에 대한 고려도 필요하다. 요구 페이징이 실제 휴대 단말기에서 사용되는 예는 적지만 향후 사용될 가능성이 높을 것으로 예상되므로 이를 플래시 메모리에 적합하도록 최적화할 필요가 있다.

플래시 메모리 및 DRAM 상에 파일시스템을 어떤 크기로 어떻게 구성하는 것이 최적인지에 대한

연구가 필요하다. 루트 파일 시스템과 사용자 파일 시스템을 구별하고 각 파일 시스템에 저장되어야 하는 파일들의 종류 및 각 파일들의 수정 가능성 등을 고려하여 파일 시스템 구성을 결정해야 한다. 또한, 시스템 운용 중 파일 시스템 구성 파일들의 변동 가능성을 고려하여 어떤 파일 시스템을 쓰기 가능 파일 시스템으로 구성할지 결정해야 한다.

이 밖에도 휴대 단말에 탑재되는 장치의 종류가 많아지면서 이들을 효과적으로 지원하기 위한 연구가 필요하는 등 휴대 단말의 발전에 따라 고려해야 하는 사항들은 계속 늘어나고 있다.

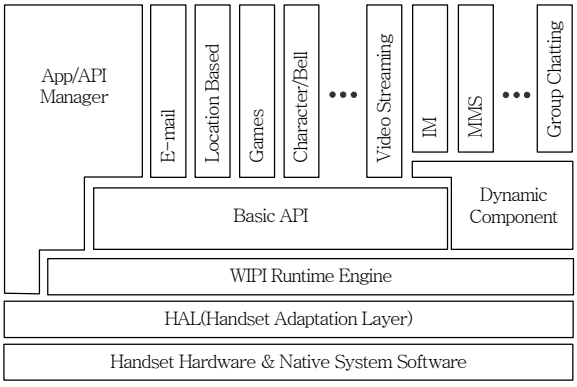
Ⅲ. 모바일 소프트웨어 플랫폼 동향

지금까지는 스마트폰을 대상으로 하는 모바일 플랫폼을 개발할 때 고려해야 하는 사항들을 다루었다. 이 장에서는 현재 논점이 되고 있는 스마트폰용 모바일 플랫폼의 특징과 동향을 살펴본다.

1. WIPI

WIPI는 무선인터넷 환경을 위한 플랫폼으로 과거 국내 이동통신사별로 GVM, SK-VM, MAP, BREW, CLDC/MIDP 등으로 분화되어 있던 모바일 플랫폼을 하나로 표준화한 무선인터넷 표준 플랫폼이다[7],[8]. WIPI는 이동통신 사업자, 단말기 제조사, 그리고 플랫폼 개발 업체들이 중심이 되어 만든 공개 표준이라는 특징을 가지고 있다. 소스 레벨의 호환성을 지향하는 WIPI는 플랫폼 표준화를 통하여 콘텐츠 제작 업체의 부담을 크게 줄였으며, 이동통신 사업자들도 다양한 콘텐츠를 빠르게 사용자에게 제공할 수 있게 되었다. WIPI는 2001년 9월 이동통신 3사의 공통 RFP 발표를 시작으로 2002년 3월에는 v1.0을 발표하였고, 2005년 8월 v2.0.2를 발표하였다. 현재는 범용 운영체제를 탑재한 휴대형 단말을 위해 WIPI 규격 3.0을 준비중에 있다.

(그림 5)는 현재 WIPI의 구조도이다. 핸드셋 소프트웨어 위에 HAL 계층이 있어 핸드셋 소프트웨어와 WIPI 플랫폼의 포팅을 도와준다. HAL 계층의 역할은 다양한 핸드셋 환경에서도 HAL이 요구하는 기본적인 API만 충족시키면 WIPI를 탑재할 수 있도록 하기 위함이다. WIPI의 핵심이라고 할 수 있는 WIPI Runtime Engine은 WIPI 콘텐츠를 동작시키는 주체다. J2ME의 경우 응용프로그램이 바이트코드로 컴파일됨에 비해 WIPI 응용프로그램은 바이너리로 컴파일되는데, WIPI Runtime Engine은 바이너리 형식의 WIPI 응용프로그램을 실행시키고 관리한다. WIPI 응용프로그램에 노출되는 Basic API는 C/Java API로 다양한 서비스가 가능하도록 풍부한 API를 제공한다. 2002년 8월부터는 Sun의 MIDP 프로파일을 Java API에 포함시켜 공인받은 API를 제공하는 기능적인 플랫폼이 되었다. 핸드셋의 발전으로 인하여 다양한 서비스가 필요하나 Basic API가 이를 지원하지 못하는 경우 dynamic compo-



(그림 5) WIPI 플랫폼 구조도

nent를 사용하여 WIPI의 기능을 확장하여 사용할 수 있다.

WIPI의 초기 탄생은 기존의 이동통신 단말기에서 시작하였다. 그러나, 텔레매틱스, DMB, RFID 등 다양한 분야에서 WIPI를 차용하여 사용하고자 하는 요구가 늘어 WIPI의 규격을 제안할 수 있는 열린 창구인 WSP(WIPI Standardization Process)를 제정하여 진행하고 있다. 현재는 WIPI3.0을 통해 다양한 단말과 범용 운영체제에 대응할 수 있는 규격을 준비중에 있다.

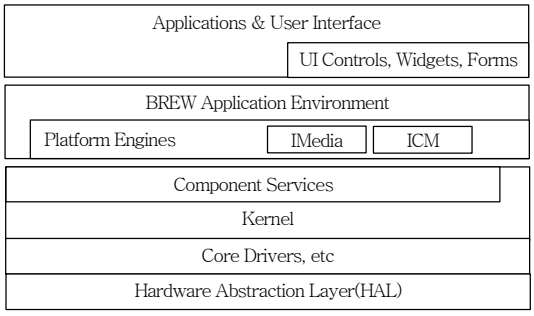
2. BREW

BREW는 핸드셋용 응용프로그램 개발을 위한 표준 개발 환경을 제공한다. 핸드셋 칩 제조 업체인 Qualcomm에서 개발한 플랫폼으로 빠른 개발 주기와 기능 개선과 확장이 빠르다는 장점을 가지고 있다. 하위 운영체제의 소프트웨어에 대한 지식 없이 BREW에서 제공하는 풍부한 API를 쉽게 응용프로그램이 가능하며, BREW 응용프로그램은 바이너리로 동작하므로 빠른 실행이 가능하다는 장점을 가지고 있다[9].

BREW는 핸드셋에서 데스크톱과 유사한 환경을 제공하여 사용과 개발이 용이하다는 장점을 가지고 있다. 다른 윈도 시스템들과 같이 이벤트에 기반한 구조를 가지고 있고, C 언어를 통해 개발하며, 개발사 및 제조사에 의해 부족한 기능을 확장할 수 있는 등 풍부한 편의를 제공하고 있다[10].

Qualcomm의 uiOne은 개선된 UI 프레임워크로 커스터마이징과 퍼스널라이징이 가능한 프레임워크다. BREW의 extension으로 'Data is the UI' 개념을 사용하여 UI를 play 할 수 있는 데이터로 표현한다[11]. 이를 위해 TrigML을 UI를 표현하기 위한 마크업 언어로 사용하였다. TrigML은 Trigenix의 기술로 2004년 10월 Qualcomm에서 회사를 인수하면서 uiOne 기술로 포함되었다.

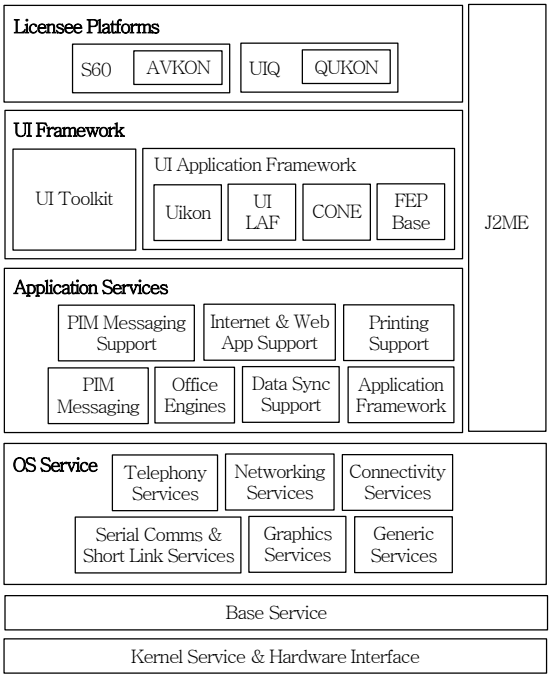
Qualcomm은 (그림 6)과 같은 BREW 4 시리즈의 디자인을 공개하면서 BREW 4 시리즈가 컴포넌트에 기반한 플랫폼임을 밝혔다[12]. 핸드셋 소프트웨어 위에 컴포넌트 서비스 환경을 두었다. 컴포넌트 서비스 환경은 파일 시스템, 컴포넌트 매니저, 스레드 및 프로세스 처리, 메모리 관리 등과 같이 휴대 단말에서 공통으로 요구하는 컴포넌트를 기본적으로 탑재하고 기능 확장을 위해 컴포넌트를 추가하는 형식으로 플랫폼을 확장할 수 있는 구조를 제공한다. BREW 4 시리즈에 처음 도입된 컴포넌트 서비스를 통해 Qualcomm은 범용 운영체제 환경에 플랫폼을 제공하고자 한다.



(그림 6) BREW4 시리즈 플랫폼 구조도

3. Symbian

Symbian은 유럽의 주요 단말 제조 업체들의 공동 투자로 만들어진 스마트폰을 위한 운영체제로서 Nokia Series 60 스마트폰 플랫폼의 기반을 이룬다[13],[14]. Symbian 운영체제는 기본적으로 마이크로 커널 기반의 경량 운영체제 구조를 갖추고 있고 네트워크 기능을 포함한 대부분의 기능이 마이크로 커널 상의 서버의 형태로 존재한다. Symbian 운영체제의 잘 정의된 OEM adaptation layer는 하드웨어 구성 요소에 대한 지원 소프트웨어 및 통신 프로토콜 부분을 지원하기 위한 인터페이스 소프트웨어 부분 등이 서버 모듈의 형태로 정의되어 있어 다른 부분에 대한 고려 없이 비교적 독립적으로 새로운 하드웨어에 Symbian을 탑재할 수 있도록 돕는다. Symbian은 C++를 기본 언어로 하고 있으며, OPL, Python 등의 언어도 지원하며 최근엔 J2ME도 지원한다. (그림 7)은 체계적인 구조를 갖춘 Symbian의 구조도를 보여준다.



(그림 7) Symbian 구조도

(그림 7)에서 보는 바와 같이 Symbian은 S60과 UIQ의 플랫폼을 Symbian OS에 탑재한 구조를 가진다. Symbian OS는 OS service, application service, UI framework를 통해 다양한 기능을 제공한다. Symbian은 각종 표준 규격을 빠르고 완벽히 탑재하는 안정적인 플랫폼으로 Web SVG Tiny, AJAX, Flash Lite 2.0 등을 탑재한 브라우저를 통해 Web 2.0을 준비하고 있다.

4. iPhone

Apple의 iPhone은 소프트웨어적인 특징보다 버튼이 없는 터치스크린으로만 동작하는 폰으로 관심을 끌었다. 멀티 터치스트린이 가능한 480×320 해상도의 3.5인치 LCD는 두 손가락을 사용한 다양한 UI를 가능하게 하고, 가속도계와 접근탐지 센서를 활용하여 편의 기능을 제공한다. iPhone은 MAC OS X v10.4.10을 사용한다. 이 플랫폼은 FreeBSD 기반의 Darwin을 바탕으로 풍부한 멀티미디어 기능과 그래픽스 기능을 제공한 UI를 내세우고 있다.

현재 iPhone은 web application만 개발이 가능한 형태다[15]. 2008년 2월경에 native application 개발이 가능한 SDK를 제공할 예정에 있으며 native application은 objective-C 기반이 될 것으로 예상한다[16]. Web application은 Safari 엔진을 통한 web 2.0과 AJAX application을 지원하여, 누구든지 web 2.0을 통해 web application을 개발할 수 있도록 한다.

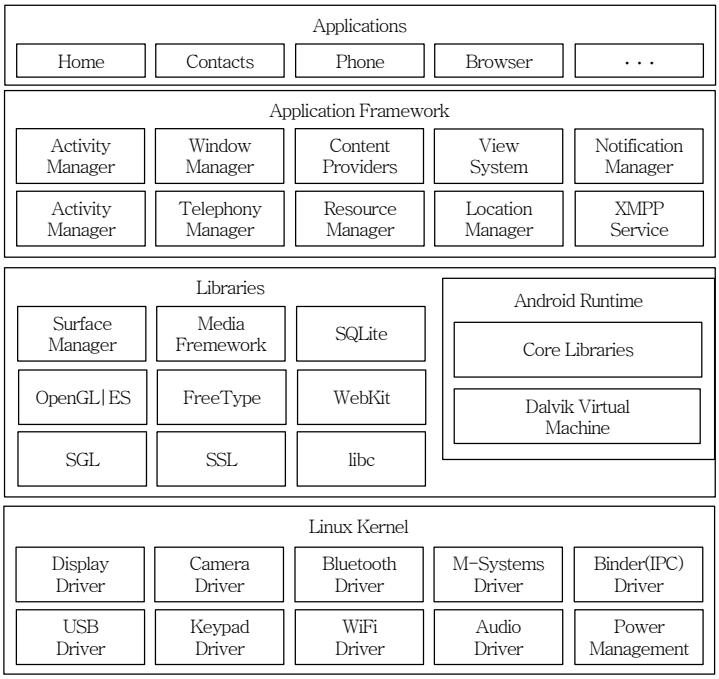
5. Android

Android 플랫폼은 모바일 디바이스를 위한 소프트웨어 스택으로 운영체제, 미들웨어를 비롯한 필수 응용프로그램을 포함하고 있다[17],[18]. Android SDK에는 개발에 필요한 기본적인 tool을 포함하며 API를 제공하여 Java 언어를 이용하여 Android용 응용프로그램을 개발할 수 있도록 되어 있다. <표 1>에서 Android 플랫폼의 주요 특징을 정리하였다.

Google이 발표한 Android 플랫폼은 (그림 8)과 같은 구조를 가지고 있다. 필수 응용프로그램들은 application 레이어에 기본적으로 탑재되어 있고, 모든 응용프로그램은 Java로 작성되었다.

〈표 1〉 Android 플랫폼 주요 특징

Feature	Description
응용 프레임워크	컴포넌트를 교체하고 재사용할 수 있음
Dalvik VM	모바일 디바이스에 최적화
통합 브라우저	WebKit 기반
최적화된 Graphics	2D 라이브러리와 OpenGL|ES 1.0[19] 라이브러리 제공
SQLite[20]	데이터 베이스
Media 지원	이미지, 오디오, 비디오(MPEG4, H.264, MP3, AAC, AMR, JPG, PNG, GIF)
GSM Telephony	하드웨어 의존적
Bluetooth, EDGE, 3G, and WiFi	
카메라, GPS, 나침반, 가속기	
풍부한 개발환경	에뮬레이터, 디버거, 프로파일링, Eclipse 플러그인



(그림 8) Android 구조도

이 응용프로그램들은 application framework가 제공하는 기능들을 사용하여 개발할 수 있는데, 컴포넌트 형태로 교체할 수 있도록 디자인 되었다. 응용프로그램들과 application framework가 Java로 구현되어 있는 것에 반해 아래 계층은 C/C++로 구현되어 있다. Library는 application framework를 통해 응용프로그램에 기능을 제공하는 구조로 되어 있다. Android Runtime은 Java 실행환경으로 Dalvik VM을 기본으로 하고 있다. Dalvik VM은 디바이스에서 복수의 VM 인스턴스를 동작시킬 수 있도록 개발되었다. 복수의 응용프로그램을 동작시키면 복수의 VM 인스턴스들이 생성되는 형식이다. Android는 Linux kernel을 채택하고 있는데, 버전 2.6에서 동작하며 보안, 메모리 관리, 프로세스 관리, 네트워크 스택, 드라이버 모델 등을 따른다[14].

Google은 Android용 응용프로그램을 쉽게 개발할 수 있도록 Eclipse용 IDE를 내놓았다. 이를 이용하면 skeleton 코드를 자동으로 생성하여 응용프로그램 개발을 쉽게 할 수 있다. 또한 개발한 응용프로그램은 Andorid 에뮬레이터에서 동작시킬 수 있다.

6. Windows Mobile

Windows Mobile은 다양한 제품군으로 다양한 디바이스를 지원한다. 최근 발표된 Windows Mobile 6.0은 C# 기능을 강화하여 더 빠른 응용프로그램 개발을 가능하게 하였다[16]. Windows Mobile 계열은 C, C++, C#을 지원하며 데스크톱에 탑재되는 Windows와 상당부분 동일한 API를 제공하여 쉽고 빠른 개발 환경을 제공한다[21]. Symbian과 마찬가지로 오랫동안 사용되어 안정성과 성능이 검증된 플랫폼으로 기본 라이브러리부터 UI까지 풍부한 기능을 제공하는 것이 특징이다.

Ⅳ. 결론

모바일 단말은 싱글 칩 또는 듀얼 칩을 탑재하여 과거 PDA의 성능을 능가하는 방향으로 나아갈 것이다. 최근 iPhone의 출시와 Android의 발표로 인해 이러한 발전 방향은 더욱 빠르게 진행될 것으로 예상된다. 퍼스널 컴퓨터 부분에서 최강자였던 Mi-

crosoft 역시 Android와 iPhone을 경계하며 Windows Mobile을 강화하려는 의지 역시 이러한 발전 방향에 동의하고 있음을 암시한다.

과거 모바일 플랫폼과 가까운 미래의 모바일 플랫폼의 가장 큰 차이는 하위 운영체제에 있다. 과거 모바일 플랫폼은 RTOS 상에 비교적 간단한 플랫폼을 위치시켰으나, 새로운 모바일 플랫폼들은 범용 운영체제 상에 플랫폼을 위치시키게 된다. 범용 운영체제가 기존의 RTOS와 다른 점은 메모리 프로텍션이나 멀티태스킹과 같은 퍼스널 컴퓨터에서 사용되던 운영체제 서비스를 제공한다는 점이다. 과거 모바일 플랫폼에서는 응용프로그램을 위해 간단한 메모리 프로텍션이나 스레드 등을 제공하였다. 그러나 범용 운영체제를 사용하면 이런 복잡하고 시스템 성능에 큰 영향을 주는 서비스들을 범용 운영체제에 일임하고 모바일 플랫폼은 순수 플랫폼의 역할을 제공할 수 있다.

WIPI를 비롯한 모바일 플랫폼은 신규 무선통신 서비스의 등장으로 인해 다양한 무선 통신 서비스를 지원할 수 있어야 한다. 다양한 멀티미디어 서비스와 Mobile RFID와 같은 신규 서비스를 지원하기 위해 플랫폼의 기능을 확대하고 표준화를 진행하여야 한다[22].

하위 운영체제가 범용 운영체제로 변화함에 따라 운영체제가 제공하는 다양한 기능을 활용할 수 있도록 HAL과 같은 플랫폼의 포팅 레이어에 대한 재정의가 필요하다. 이를 통해 다양한 기능을 제공하는 범용 운영체제 상에서 현재의 HAL 계층을 에뮬레이션 해야 하는 부자연스러운 상황을 피해야 한다.

참 고 문 헌

[1] Bruce Brown, "Microsoft's Smartphone Exceeds Expectations," http://www.pcmag.com/print_article/0,3048,a=113391,00.asp, Dec. 30, 2003.

[2] Min-Hong Yun, Do-Hyung Kim, and Sun-Ja Kim, "Experience of Linux and GTK+ 2 Smartphone," *IEE 3G2004 Conference*, London, Oct. 2004.

[3] 배준현, "차세대 운영체제 플랫폼, L4/Iguana," 마이크로소프트웨어, 2007. 1., pp.228-233.

[4] Justin Helmig, "Developing Core Software Technologies for TI's OMAP Platform," TI, 2002. 8.

[5] R. Karavets and P. Krishan, "Power Management Techniques for Mobile Communication," *The 4th Annual ACM/IEEE Int'l Conf. on Mobile Computing and Networking(MOBICOM-98)*, Oct. 25, 1988.

[6] Chun-Chieh Lin, Chuen-Liang Chen, and Ching-Hsu Tseng, "Source Code Arrangement of Embedded Java Virtual Machine for NAND Flash Memory," *Int'l Symp. on Communications and Information Technologies*, 2007.

[7] Mobil Standard Platform, TTAS.KO-06.0036, May 2002.

[8] Sun Microsystems, "Korea's Wireless Internet Standardization Forum Embraces Java Technology," http://www.sun.com/smi/Press/sunflash/2003-04/sunflash.20030428.1.html, Apr. 28, 2003.

[9] BREW Official Web Site, http://brew.qualcomm.com/brew/en/

[10] BREW 3.1 SDK, https://brewx.qualcomm.com/brew/sdk/download.jsp?page=dx/en/brew31/ad/tl/sdk

[11] Stefan Butlin, "uiOne: Developing the Core UI," *BREW Conference*, 2005.

[12] Ramesh Chandrasekhar, "Preparing for Next Generation BREW," *BREW Conference*, 2007.

[13] Symbian Developer Network, http://developer.symbian.com/

[14] Forum Nokia, http://forum.nokia.com/

[15] iPhone Dev Center, http://developer.apple.com/iphone/devcenter/

[16] 홍종진, "글로벌 하이엔드 모바일 플랫폼 동향," 2007 KWSF & WIDEF 모바일 기술 컨퍼런스 프로시딩, 2007.

[17] Open Handset Alliance, http://www.openhandsetalliance.com/

[18] Android SDK, http://code.google.com/android/

[19] Official OpenGL|ES Web site, http://www.khrnos.org/opengles/index.html

[20] SQLite: An Embeddable SQL Database Engine, http://www.sqlite.org

[21] Windows Mobile Devleoper Center, http://msdn2.microsoft.com/ko-kr/windowsmobile/default(en-us).aspx

[22] 노무라종합연구소, "2010 IT 로드맵," 매일경제신문사, 2007.